数理解析研究所講究録
992巻 1997年 167-174

λ_C 計算と λ_P 計算との対応

廣川 佐千男 (九大) 亀山 幸義 (京大) 馬場 謙介 (九大)
Sachio Hirokawa, Yukiyoshi Kameyama, Kensuke Baba

1 はじめに

Curry-Howard の対応 [9] とは、論理と型付ラムダ計算との関係であり、式は型に、証明は項にそれぞれ対応しているというものである。古典論理の構成的な説明はできないと思われていたので、この対応は直観主義に限られていた。しかし、1990年に T.Griffin [6] は、Felleisen [4, 5] の C 演算子に矛盾なく型を当てはめ、それが二重否定の除去の公理 $\neg\neg\alpha \to \alpha$ に対応していることを示した。これにより古典論理の証明に計算論的意味が与えられた。その後、古典論理に対応する型付ラムダ計算が提案されその計算論的意味が調べられている [1, 2, 3, 7, 8, 10, 11, 12, 13, 14, 15, 16, 17, 18]。

Griffin の C 演算子の型システムの発見は画期的だったが、彼の型体系には少なくとも2つの不明な点があった。1つは、最小論理 (直観主義論理から矛盾の公理 $\bot\to \alpha$ を取り除いたもの) に二重否定の除去の公理を付け加えると古典論理になることである。そこで、我々は二重否定の除去の公理の代りに、矛盾の公理 $\bot\to \alpha$ と Peirce の公理 $((\alpha \to \beta) \to \alpha) \to \alpha$ を用いる。これにより、古典論理において、直観主義論理によるものと純粋に古典論理よるものが下図のように区別できるようになる。

$$\begin{aligned} \text{古典論理} &= \text{最小論理} + \qquad \text{二重否定の除去の公理} \\ &= \text{最小論理} + \text{矛盾の公理} + \text{Peirceの公理} \\ &= \qquad \text{直観主義論理} \qquad + \text{Peirceの公理} \end{aligned}$$

2つ目の点は、Felleisen の計算規則では、項全体の型が $\bot$ でなければならないという点である。ところが型 $\bot$ を持つ項は有り得ないから、この計算は適用できない。

$$E[CN] \ \to \ N(\lambda x.A(E[x]))$$

この問題は Griffin と Felleisen による解も知られている。我々は、次のような計算規則をもつ call/cc 演算を用いて、この問題を避ける。

$$E[call/ccN] \ \to \ E[N(\lambda x.A(E[x]))]$$

ここで E[call/cc N] は任意の型 β となることができ、call/cc の型は $((\alpha \to \beta) \to \alpha) \to \alpha$ すなわち Peirce の公理になっている。

本論文ではこの2つの公理に対応する型付結合子論理の拡張として、2つの結合子論理の計算体系、λPJ と λC を定義する。ここで導入される結合子 P, J, C はそれぞれ、Peirce の公理、

矛盾の公理、二重否定の除去の公理を型として持つ。ラムダ計算における β、η の他に新たに論理計算、簡単化、空計算、擬 η 計算の 4 種類の計算規則を導入する。それぞれの計算規則は、文献にある古典論理の計算に深く関わっている。例えば、Felleisen の C_L は、我々の簡単化である。我々は λPJ と λC の間の変換を定義し、一方での計算規則が他方での計算規則で模倣できることを示す。ただし、P に関する計算では、P の型を制限すれば対応がつくが、一般の形では対応がつくかどうか、まだ未解決の部分もある。

我々はまた、$\overline{\lambda}$J と $\underline{\lambda}$ という型付きラムダ計算の拡張を定義する。$\overline{\lambda}J$ は、結合子 P, J に対応し、$\underline{\lambda}$ は、結合子 C に対応する。つまり、このラムダ計算は通常の結合子論理とラムダ計算の対応の自然な拡張になっている。この 2 つの拡張したラムダ項は結合子よりも簡潔な表現が得られる。

この論文の要点は、(1) 古典論理の推論規則に対応する結合子を持つ 2 つの計算体系を与えた事、(2) 計算規則のさまざまなクラスを分析した事、(3) 結合子論理とラムダ計算の対応を示したことである。

2 古典論理に対応する結合子論理

この章では純粋な型付ラムダ計算の拡張として、2 つの計算体系 λPJ、λC を導入しその間の関係を考える。

この論文での計算の基本となっているのは、最小論理の含意切片である。含意切片は様々な論理の本質であり、この性質を理解すれば、それを他の結合記号に取り入れるのは容易である。

型は次のように定義される。ただし、否定は $\neg\alpha = \alpha \to \bot$ で定義する。

$$\alpha ::= A \mid \bot \mid \alpha \to \alpha$$

λC、λP の擬項は次のように定義される。ここで、x^α は型が α である変数を表す。

$$\begin{array}{ll} (;\lambda C\text{について}) & M ::= x^\alpha \mid (\lambda x^\alpha.M) \mid (MM) \mid C \\ (\lambda PJ\text{について}) & M ::= x^\alpha \mid (\lambda x^\alpha.M) \mid (MM) \mid P \mid J \end{array}$$

M の自由変数の集合を $FV(M)$ と表記する。また、x^α の α を省略することがある。

通常のラムダ項の型システム

$$\frac{}{x^\alpha : \alpha} \qquad \frac{M : \beta}{\lambda x^\alpha.M : \alpha \to \beta} \qquad \frac{M : \alpha \to \beta \qquad N : \alpha}{MN : \beta}$$

新しい結合子の型

$$\frac{}{C : \neg\neg\alpha \to \alpha} \qquad \frac{}{P : ((\alpha \to \beta) \to \alpha) \to \alpha} \qquad \frac{}{J : \bot \to \alpha}$$

項とは、上の型システムによって型付けされた擬項で、型は M^α のように項の右上に表記することがある。代入は $M[x := N]$ のようにかく。P について、型が $((\alpha \to \bot) \to \alpha) \to \alpha$ と

なっているとき、$P_\perp$ とかく。λC と λPJ の文法の定義は以上である。計算規則については後で示す。

次の定理はよく知られている。

定理 **1** α を型とすると、次の3つは同値である。
(1) α は古典論理で証明可能である。
(2) λC で $M:\alpha$ が証明可能な閉じた λC-項 M が存在する。
(3) λPJ で $M:\alpha$ が証明可能な閉じた λPJ-項 M が存在する。

この定理は、2つの計算が論理学的に古典論理に等しいことを示している。

2.1 計算規則

通常のラムダ計算における計算規則は β と η の2つである。

$$\begin{array}{lcll}(\lambda x^\alpha.M^\beta)N^\alpha & \to & M^\beta[x^\alpha := N^\alpha] & (\beta)\\ \lambda x^\alpha.M^{\alpha\to\beta}x^\alpha & \to & M^{\alpha\to\beta} & (\eta)\end{array}$$

ただし η-reduction については、$x \notin FV(M)$ である。

λC ならびに λPJ には、この他に次のような4つの計算のクラスがある。
論理計算規則　1つ目のクラスは、廣川らの P- 計算 [7] に端を発するもので、論理計算規則と呼ぶ。

$$\begin{array}{lcll}M^{\neg\alpha}(C^{\neg\neg\alpha\to\alpha}N^{\neg\neg\alpha}) & \to & N^{\neg\neg\alpha}M^{\neg\alpha} & (\mathtt{C})\\ M^{\alpha\to\beta}(P^{((\alpha\to\beta)\to\alpha)\to\alpha}N^{(\alpha\to\beta)\to\alpha}) & \to & M^{\alpha\to\beta}(N^{(\alpha\to\beta)\to\alpha}M^{\alpha\to\beta}) & (\mathtt{P})\\ M^{\neg\alpha}(J^{\perp\to\alpha}N^{\perp}) & \to & N^{\perp} & (\mathtt{J})\end{array}$$

これらの計算規則は型を保ち、証明図の変換とみなしたとき自然であることが分かる。しかし、C や P の計算規則を型なしの項についての計算規則として使うと、たちどころに合流性が成り立たなくなる。他の計算体系との対応を簡単にするため、関数適用文脈 (applicative context) を用いた論理計算規則の別の表現を導入する。関数適用文脈とは、穴 [ ] がラムダ抽象の中に現われていない文脈である:

$$E[\,] ::= [\,] \mid (E[\,]M) \mid (ME[\,])$$

これを使った論理計算規則の定義は、下のようになる。

$$\begin{array}{lcll}E^\perp[C^{\neg\neg\alpha\to\alpha}N^{\neg\neg\alpha}] & \to & N^{\neg\neg\alpha}(\lambda x^\alpha.E^\perp[x]) & (\mathtt{EC})\\ E^\beta[P^{((\alpha\to\beta)\to\alpha)\to\alpha}N^{(\alpha\to\beta)\to\alpha}] & \to & E^\beta[N^{(\alpha\to\beta)\to\alpha}(\lambda x^\alpha.E^\beta[x])] & (\mathtt{EP})\\ E^\perp[J^{\perp\to\alpha}N^\perp] & \to & N^\perp & (\mathtt{EJ})\end{array}$$

論理計算規則の2つの表現は、β、η 計算を無視すると等しい。後者による表現でも合流性は成り立たないが、そうなるような部分計算は考えやすい。

簡単化の計算規則　2つ目のクラスは、簡単化の計算規則で、次のように定義される。型の表示は省略した。

$$
\begin{array}{lcll}
(CM)N & \to & C(\lambda z.M(\lambda u.z(uN))) & (\mathtt{Csimp}) \\
(PM)N & \to & P(\lambda z.M(\lambda u.z(uN))N) & (\mathtt{Psimp}) \\
PM & \to & P_\perp(\lambda z.M(\lambda u.J(zu))) & (\mathtt{P}_\perp\mathtt{simp}) \\
(JM)N & \to & JM & (\mathtt{Jsimp})
\end{array}
$$

空計算は、次のように定義される。

$$
\begin{array}{lcll}
C^{\neg\neg\perp\to\perp}(\lambda x^{\neg\perp}.M^\perp) & \to & M^\perp & (\mathtt{C}_0) \\
P^{((\alpha\to\beta)\to\alpha)\to\alpha}(\lambda x^{\neg\alpha}.M^\alpha) & \to & M^\alpha & (\mathtt{P}_0) \\
J^{\neg\perp}M^\perp & \to & M^\perp & (\mathtt{J}_0)
\end{array}
$$

上の 2 つの計算規則 $(C_0), (P_0)$ については、$x \notin FV(M)$。

擬 η 計算規則は次のように定義される。

$$
\begin{array}{lcll}
C^{\neg\neg\alpha\to\alpha}(\lambda x^{\neg\alpha}.xM^\alpha) & \to & M^\alpha & (\mathtt{C}_\eta) \\
P^{(\neg\alpha\to\alpha)\to\alpha}(\lambda x^{\neg\alpha}.J^{\perp\to\alpha}(xM^\alpha)) & \to & M^\alpha & (\mathtt{P}_\eta) \\
C^{\neg\neg\alpha\to\alpha}(\lambda x^{\neg\alpha}.x(C^{\neg\neg\alpha\to\alpha}(\lambda y^{\neg\alpha}.xM^\alpha))) & \to & M^\alpha & (\mathtt{C}_\Delta)
\end{array}
$$

ただし、$x \notin FV(M)$。

2.2 計算の模倣

この節では、4 つの計算規則のクラスについて、λPJ と λC の関係を考える。まづ、λC から λPJ への変換を次のように定義すると次の定理が成り立つ。

$$
\begin{array}{rcl}
x^\circ & \to & x \\
(\lambda x.M)^\circ & \to & \lambda x.M^\circ \\
(MN)^\circ & \to & M^\circ N^\circ \\
C^\circ & \to & \lambda x.P(\lambda y.J(xy))
\end{array}
$$

定理 2 (1) λC で $M:\alpha$ ならば、λPJ で $M^\circ:\alpha$ である。
(2) (EC) は、(EP) と (EJ) で模倣できる。
(3) (Csimp) は、(Psimp) と (Jsimp) で弱い意味で模倣できる。
(4) $(\mathtt{C}_0)$ は、$(\mathtt{P}_0)$ と $(\mathtt{J}_0)$ で模倣できる。
(5) $(\mathtt{C}_\eta)$ は、$(\mathtt{P}_\eta)$ で模倣できる。
(6) $(\mathtt{C}_\Delta)$ は、(J) と $(\mathtt{P}_\eta)$ で模倣できる。

λPJ から λC への変換 (その 1)

λPJ から λC への変換には (EC) は (EP) に比べて型が制限されているという問題がある。C は P に対応しているのではなく、$P_\perp$(と J) に対応していると考えられる。そこで、λPJ から λC への変換を 2 つ定義する。1 つは $\lambda\mathrm{P}_\perp\mathrm{J}$ から λC への変換、もう 1 つは λPJ すべてから λC への変換である。これは $\lambda\mathrm{P}_\perp\mathrm{J}$ から λC への変換である。

$$
\begin{array}{rcl}
x^\bullet & \to & x \\
(\lambda x.M)^\bullet & \to & \lambda x.M^\bullet
\end{array}
$$

$$\begin{aligned}(MN)^\bullet &\to M^\bullet N^\bullet\\ P_\perp^\bullet &\to \lambda x.C(\lambda y.y(xy))\\ J^\bullet &\to \lambda x.C(\lambda y.x)\end{aligned}$$

定理 **3** $\bullet$ について次の *(1)-(6)* が成り立つ。
(1) $\lambda P_\perp J$ で $M:\alpha$ ならば、λC で $M^\bullet:\alpha$ である。
(2) (EP) と (EJ) は (EC) で模倣できる。
(3) (Psimp) と (Jsimp) は (Csimp) で弱い意味で模倣できる。
(4) (P_0) は (C_η) で模倣できる。
(5) (J_0) は (C_0) で模倣できる。
(6) (P_η) は (C_Δ) で模倣できる。

λC から λPJ への変換 (その 2)

λPJ から λC への変換について、次の P の変換を与える。

$$P^\bullet = \lambda x.C(\lambda y.y(x(\lambda z.C(\lambda u.yz))))$$

他はその 1 の変換と同じである。

定理 **4** 変換 $\bullet$ について次の *(1)-(7)* が成り立つ。(1) λPJ で $M:\alpha$ ならば λC で $M^\bullet:\alpha$ ある。
(2) (EJ) は (EC) で模倣できる。
(3) (Psimp) と (Jsimp) は (Csimp) で弱い意味で模倣できる。
(4) ($P_\perp$simp) は (Csimp) と (C_0) で弱い意味で模倣できる。
(5) (P_0) は (C_η) で模倣できる。
(6) (J_0) は (C_0) で模倣できる。
(7) (P_η) は (C_Δ) と (C_0) で模倣できる。

上の 2 つの定理をまとめると下のような図となる。ここで、$\Leftrightarrow$ は一方の計算規則をもう一方の計算規則で模倣できるということ。$\leftrightarrow$ は弱い模倣を意味している。ここでは、λPJ ではなく、$\lambda P_\perp J$ と λC の関係である。

$$\begin{array}{rcl}\lambda C & & \lambda P_\perp J\\ (EC) & \Leftrightarrow & (EP)+(EJ)\\ (Csimp) & \leftrightarrow & (Psimp)+(Jsimp)\\ (EC)+(Csimp) & \leftrightarrow & (EP)+(EJ)+(Psimp)+(Jsimp)\end{array}$$

空計算規則と擬 η 計算規則を含む対応については良い結果は得られなかった。また、λPJ と λC との変換では (Csimp) $\leftrightarrow$ (Psimp) + (Jsimp) だけが意味のある対応になっている。

3 古典論理に対応するラムダ計算

この章では、$\underline{\lambda}$ と $\overline{\lambda}$J という 2 つの計算体系を与える。この 2 つの計算は結合子を使わないラムダ計算の形をしている。$\underline{\lambda}$ は λC に、$\overline{\lambda}$J は λPJ にそれぞれ対応している。

まず擬項を定義する。

$$(\underline{\lambda}\text{について}) \quad M ::= x \mid (\lambda x.M) \mid (MM) \mid (\lambda \underline{x}.M)$$
$$(\overline{\lambda}J\text{について}) \quad M ::= x \mid (\lambda x.M) \mid (MM) \mid (\lambda \overline{x}.M) \mid J$$

$\lambda x.\lambda \overline{y}.\lambda \overline{z}.xyz$ は $\lambda x \overline{yz}.xyz$、$\lambda x.\lambda \underline{y}.\lambda \underline{z}.xyz$ は $\lambda x \underline{yz}.xyz$ のように略記する。

型システムとしてはラムダ計算の型に次のような規則を加える。

$$\frac{\begin{array}{c}[x:\neg\alpha] \\ \vdots \\ M:\bot\end{array}}{\lambda \underline{x}.M:\alpha} \qquad \frac{\begin{array}{c}[x:\alpha\to\beta] \\ \vdots \\ M:\alpha\end{array}}{\lambda \overline{x}.M:\alpha}$$

仮定 $x:\neg\alpha$、$x:\alpha\to\beta$ は、この推論により除去される。

3.1 $\underline{\lambda}, \overline{\lambda}$の 計算規則

結合子論理の場合と同様に、計算規則は 4 つに分けられる。J については前のままで、$\lambda \underline{x}.M$ と $\lambda \overline{x}.M$ についてのみ定義する。

論理計算規則

$$E[\lambda \underline{x} M] \to M[x := \lambda y.E[y]] \qquad (\mathtt{EC})$$
$$E[\lambda \overline{x} M] \to E[M[x := \lambda y.E[y]]] \qquad (\mathtt{EP})$$

簡単化の計算規則

$$(\lambda \underline{x}.M)N \to \lambda \underline{z}.M[x := \lambda u.z(uN)] \qquad (\mathtt{Csimp})$$
$$(\lambda \overline{x}.M)N \to \lambda \overline{z}.M[x := \lambda u.z(uN)]N \qquad (\mathtt{Psimp})$$
$$\lambda \overline{x}.M \to \lambda \overline{z}.M[x := \lambda u.J(zu)] \qquad (\mathtt{P}_\bot\mathtt{simp})$$

空計算規則

$$\lambda \underline{x}.M \to M \qquad (\mathtt{C_0})$$
$$\lambda \overline{x}.M \to M \qquad (\mathtt{P_0})$$

ただし $x \notin FV(M)$。

擬 η 計算規則

$$\lambda \underline{x}.xM \to M \qquad (\mathtt{C}_\eta)$$
$$\lambda \overline{x}.J(xM) \to M \qquad (\mathtt{P}_\eta)$$
$$\lambda \overline{x}.x(\lambda \overline{y}.xM) \to M \qquad (\mathtt{C}_\Delta)$$

ただし $x \notin FV(M)$。

3.2 計算の模倣

結合子論理時と同様に、$\underline{\lambda}$ と $\overline{\lambda}J$ についても互いを模倣できる。$\underline{\lambda}$ から $\overline{\lambda}J$ への変換 $\circ$、逆の変換を $\bullet$ はそれぞれ次のように定義できる。

$$\begin{array}{rcl} x^{\circ} & \to & x \\ (\lambda x.M)^{\circ} & \to & \lambda x.M^{\circ} \\ (MN)^{\circ} & \to & M^{\circ}N^{\circ} \\ (\lambda \underline{x}.M)^{\circ} & \to & \lambda \overline{x}.JM \end{array} \qquad \begin{array}{rcl} x^{\bullet} & \to & x \\ (\lambda x.M)^{\bullet} & \to & \lambda x.M^{\bullet} \\ (MN)^{\bullet} & \to & M^{\bullet}N^{\bullet} \\ (\lambda \overline{z}.M)^{\bullet} & \to & \lambda \underline{z}.zM \\ J^{\bullet} & \to & \lambda x\underline{y}.x \end{array}$$

この変換によって、定理 2、3、4 と同様の結果を得る。

3.3 結合子とラムダ計算の対応

結合子論理の λC、λPJ と、ラムダ計算の $\underline{\lambda}$、$\overline{\lambda}J$ はそれぞれ対応している。結合子からラムダ項への変換 $\sharp$、逆の変換 $\flat$ は次のように定義できる。

$$\begin{array}{rcl} x^{\sharp} & \to & x \\ (\lambda x.M)^{\sharp} & \to & \lambda x.M^{\sharp} \\ (MN)^{\sharp} & \to & M^{\sharp}N^{\sharp} \\ C^{\sharp} & \to & \lambda x\underline{y}.xy \\ P^{\sharp} & \to & \lambda x\overline{y}.xy \\ J^{\sharp} & \to & J \end{array} \qquad \begin{array}{rcl} x^{\flat} & \to & x \\ (\lambda x.M)^{\flat} & \to & \lambda x.M^{\flat} \\ (MN)^{\flat} & \to & M^{\flat}N^{\flat} \\ (\lambda \underline{x}.M)^{\flat} & \to & C(\lambda x.M^{\flat}) \\ (\lambda \overline{x}.M)^{\flat} & \to & P(\lambda x.M^{\flat}) \\ J^{\flat} & \to & J \end{array}$$

定理 **5** $\sharp$、$\flat$ について次の *(1)-(3)* が成り立つ。
(1) λC*(*λPJ*)* で $M:\alpha$ ならば、$\underline{\lambda}$ *(*$\overline{\lambda}J$*)* で $M^{\sharp}:\alpha$、また $\underline{\lambda}$ *(*$\overline{\lambda}J$*)* で $N:\alpha$ ならば、λC*(*λPJ*)* で $N^{\flat}:\alpha$ である。
(2) 全ての reduction は、同じ名前の reduction で模倣できる。
(3) 2 つの変換は、逆の変換になっている。つまり、全ての λC (λPJ)-term$M$、$\underline{\lambda}(\overline{\lambda}$J)-term$N$ について、$(M^{\sharp})^{\flat}=M$、$(N^{\flat})^{\sharp}=N$ である。

この定理により、結合子論理の λC、λPJ と、ラムダ計算の $\underline{\lambda}$、$\overline{\lambda}J$ はそれぞれ同じシステムとみなすことができる。

参考文献

[1] Barbanera, F. and S. Berardi: "Extracting Constructive Content from Classical Logic via Control-like Reductions", Typed Lambda Calculi and it Applications, *Lecture Notes in Computer Science* **664**, pp. 513-531, 1991.

[2] de Groote, Ph.: "On the Relation between the $\lambda\mu$-Calculus and the Syntactic Theory of Sequential Control", *Lecture Notes in Computer Science* **822**, pp. 31–43, 1994.

[3] de Groote, Ph.: "A CPS-Translation of the $\lambda\mu$-Calculus", CAAP'94, *Lecture Notes in Computer Science* **787**, pp. 85–99, 1994.

[4] Felleisen, M., D. P. Friedman, E. Kohlbecker and B. Duba: "A Syntactic Theory of Sequential Control, *Theoretical Computer Science* **52**, pp. 205–237, 1987.

[5] Felleisen, M. and R. Hieb: "The Revised Report on the Syntactic Theories of Sequential Control and State", *Theoretical Computer Science* **103**, pp. 235-271, 1992.

[6] Griffin, T.: "A Formulae-as-Types Notion of Control", Conference Record of 17th ACM Symposium on Principles of Programming Languages, pp. 47–58, 1990.

[7] Hirokawa, S., S. Komori and I. Takeuti: "A Reduction Rule for Peirce Formula", Studia Logica 56, 3, pp. 419–426, 1996.

[8] Hirokawa, S.: "Right Weakening and Right Contraction in LK", Proc. CATS'96, Australian Computer Science Communications, Vol. 18, No. 3, pp. 168–174, 1996.

[9] Howard, W. A.: "The Formulae-as-Types Notion of Construction", reprinted in *The Curry-Howard Isomorphism* (de Groote, Ph. ed.), Academia, 1995.

[10] Murthy, C.: "An Evaluation Semantics for Classical Proofs", Proc. 6th Annual IEEE Symposium on Logic in Computer Science, pp. 96–107, 1991.

[11] Nakano, H.: "A Constructive Formalization of the Catch and Throw Mechanism", *Proc. 7th Annual IEEE Symposium on Logic in Computer Science*, pp. 82-89, 1992.

[12] Nishizaki, S.: "Programs with Continuations and Linear Logic", TACS'91 Proceedings, *Lecture Notes in Computer Science* **526** (T. Ito and A. R. Meyer eds.), pp. 513-531, 1991.

[13] Ong, C.-H. L.: "A Semantic View of Classical Proofs", *Proc. 11th IEEE Symposium on Logic in Computer Science*, pp. 230–241, 1996.

[14] Ong, C.-H. L. and C. A. Stewart: "A Curry-Howard Foundation for Functional Computation with Control", *Proc. 24th ACM Symposium on Principles of Programming Languages*, 1997.

[15] Parigot, M.: "$\lambda\mu$-calculus: An Algorithmic Interpretation of Classical Natural Deduction", Proc. International Conference on Logic Programming and Automated Reasoning (A. Voronkov ed.), *Lecture Notes in Artificial Intelligence* **624**, pp. 190–201, Springer, 1992.

[16] Parigot, M.: "Classical Proofs as Programs", KGC'93, *Lecture Notes in Computer Science* **713**, pp. 263-276, 1993.

[17] Rehof, N. J. and M. H. Sørensen: "The λ_Δ-calculus", Theoretical Aspects of Computer Software, *Lecture Notes in Computer Science* **789** (M. Hagiya and J. C. Mitchell eds.), pp. 516-542, 1994.

[18] Sato, M.: "Intuitionistic and Classical Natural Deduction Systems with the Catch and the Throw Rules", *Theoretical Computer Science*, to appear.